2551 nc

# Design-Based Mission Operation[1]

Meemong Lee, Richard J. Weidner, Wenwen Lu
Jet Propulsion Laboratory
4800 Oak Grove, Pasadena, CA 90272
818-254-2228
meemong.lee@jpl.nasa.gov,
richard.weidner@jpl.nasa.gov,
wenwen.lu@jpl.nasa.gov,

Abstract --- The Virtual Mission project led by the Mission Simulation and Instrument Modeling Group at JPL has been playing an active role in the NASA-wide information technology infusion programs, such as, Information System Technology, Next-Generation Infrastructure Technology, and Intelligent Synthesis Environment. The goal of the Virtual Mission project is to enable automated design space exploration, progressive design optimization, and lifecycle-wide design validation to ensure mission success. Design-based mission operation has been a major part of the research effort in order to establish system-wide as well as lifecycle-wide impact analysis as an integral part of the mission design process. The design-based mission operation is approached by implementing Virtual Mission Lifecycle (VML), modeling and simulation tools and system engineering processes for building a virtual mission system that can perform a realistic mission operation during the design phase of a mission. As in the real mission lifecycle convention, the VML is composed of design, development, integration and test, and operation phases. This paper describes the four phases of the VML addressing a major challenge per phase, mission model framework, virtual prototyping, agent-based mission system integration, and virtual mission operation.

TABLE OF CONTENTS

## 1. INTRODUCTION

The Virtual Mission project is a consortium of modeling and simulation technology R&D activities led by the Mission Simulation and Instrument Modeling Group at JPL. The NASA technology programs that have been supporting the R&D activities include Space Science Information System (SSIS), Next Generation Infrastructure System (NGIS), High-performance Computing and Communication (HPCC), and Intelligent Synthesis Environment initiative (ISE). Each program addresses a specific technology area in modeling and simulation: SSIS, science targets and experiment scenarios; NGIS, mission model infrastructure; HPCC, distributed mission simulation, and ISE, system lifecycle synthesis. These technology areas are main ingredients for developing a dynamically configurable virtual mission system that can be operated in a wide range of hypothetical target body phenomena.

The VM project pursues the following three objectives to facilitate a revolutionary mission system design process:

1) *Reversible design:* to formulate a mission model space in which all phases of the lifecycle can be expressed sharing a common mission model taxonomy, and the design processes can be propagated bi-directionally from components to system to operation, as well as from operation to system to components.
2) *Integrated design*: to develop a virtual mission system that can simulate an integrated system behavior based on the representation of the subsystem designs, so that each subsystem design can be progressively refined with system-level perspectives and feedback.
3) *Validated design*: to perform system level operation analyses for comprehensive validation of the mission system design with realistic operation scenarios and mission environments. The system-level operation validates subsystem interactions, time-dependent resource usage, and environment sensitive command and data handling.

The VM approaches these objectives by implementing three interacting modeling and simulation layers: a mission model architecture layer, a mission system simulation layer, and a mission operation simulation layer. In a previous paper, "Mission Lifecycle Modeling and Simulation" [1], the three layers were described from the over-all mission lifecycle perspective. This paper discusses the perspective from the operation phase of the mission lifecycle with an emphasis on the third objective, validated design.

During the operation phase, the operability of the mission system and the impact of the subsystem interdependencies are comprehensively analyzed. If the analysis results are not

[1] 0-7803-5/00/$10.00 © 2001 IEEE

nc ıı

favorable, it indicates that the science return may be reduced since it will be too late to change the mission system after launch. The VM project considers a comprehensive and iterative validation of the mission system design by enabling execution of the operation phase activities during the design phase is the ultimate design optimization process.

The system-level operation analysis of the validated design requires development of a pseudo-mission system that can be operated for realistic science observation scenarios. The pseudo-mission system is referred to as the Virtual Mission System (VMS), indicating that it is an integration of subsystems that are implemented entirely in software. The subsystem categorization of VMS is logical (versus physical) in that the subsystems are created to represent the mission operation functions (navigation, attitude control, communication, observation, etc.). Each logical subsystem may be mapped to a set of hardware and software components in a real mission system.

This paper discusses the process of design, development, integration, and operation of the Virtual Mission System. This process is referred to as the Virtual Mission Lifecycle (VML). In Section 2, the technical challenges of the four phases of VML are described along with a brief summary of the technical approaches. The subsequent four sections provide detailed discussions on the technical approaches: Section 3, mission-generic modeling framework; Section 4, model-based virtual subsystem prototyping; Section 5, intelligent mission model agents for dynamic system integration; and Section 6, time-based operation simulation and distributed visualization for simulation process monitoring. The final two sections present on going and planned extensions of VM. Section 7 presents three on going efforts of applying VM to the entire mission lifecycle: development phase (Instrument Testbed), integration phase (Flight System Testbed), and operation phase (science observation design for Deep Space One mission). Section 8 introduces a new research activity, Space Mission Semiotics, where an evolutionary communication framework that can be used as a framework for intelligent spacecraft system development will be pursued as an extension to VM.

## 2. VIRTUAL MISSION LIFECYCLE

The VML is composed of four phases: property model design, virtual prototype development, agent-based distributed simulation system integration, and virtual mission operation. The relationship between the VML and the mission lifecycle is depicted in Figure 1 where the VML is introduced as a shortcut to mission success validation during the concept design and detailed design phases. The software systems in VML are implemented in C++ for UNIX platforms and in Visual C++ for Windows platforms. The VML implementation closely observes the adaptive object-oriented software engineering principle [2]. This section describes four major technical challenges in the area of modeling and simulation, one major challenge per phase, following the lifecycle order: mission-generic mission system modeling framework development, time-sensitive performance property simulation, dynamic configuration and inter-subsystem synchronization, and operation scenario design and execution monitoring. Each paragraph below introduces one of these challenges.

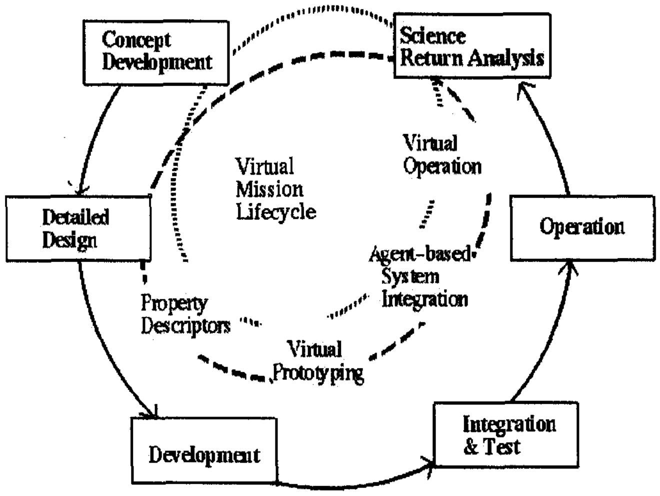


**Figure 1** Virtual Mission Lifecycle

The mission-generic modeling and simulation framework is the major challenge in the property model design phase of VML, and it is developed at two levels: a space mission context level, and a software engineering level. The mission context level deals with the functional abstraction of the mission system architecture involving logical subsystem architecture, a structured subsystem property description method, and a subsystem-domain-specific modeling language. The software engineering level, the second level of the framework, deals with adaptive object-oriented software implementation methods involving automated parser generation, automated model object class composition, dynamic instantiation of the model object, and the construction of the derived simulation class. The technical details of implementing a mission-generic mission system model framework are described in Section 3.

Time-sensitive performance property simulation refers to a high-fidelity operation behavior simulation where the operation behavior is derived by propagating the performance property at a very fine temporal resolution. In general, the high-fidelity operation behavior simulation is applied for visual verification of the integrated system behavior when the system behavior is too complex for traditional analysis. A complex system behavior example can be found during the science experiment simulation, where the science target phenomena and the operation behavior of the attitude control system, navigation system, and instrument system are integrated at a sub-second time resolution. VM achieves the time-sensitive performance property simulation by developing virtual prototypes of the subsystems. Section 4 describes the virtual prototyping process in detail.

The third phase of VML is agent-based dynamic system integration, which faces the challenges of two separate aspects of integration: the integration of mission design information, and the integration of the software processes. The information integration involves coherent interaction with a large number of distributed data sources and multidisciplinary data contents, such as spacecraft CAD files, trajectory files, target body data, subsystem performance specs, etc. The software process integration involves data-flow managers, client-server protocols, time-stamped subsystem commands, and multi-level time management. Systematic utilization of the complex design data products in a distributed mission simulation environment requires intelligent agents that are capable of understanding and interacting with their environment on the behalf of their user, of moving about the Web, and of forming and executing rudimentary decisions. Section 5 discusses implementation of three types of intelligent mission model agents that provide a dynamic mission environment configuration.

Finally, in the virtual mission operation phase of VML, autonomous operation scenario design and execution monitoring is approached by developing an event-driven observation scenario language and a distributed visualization platform. The event-driven observation scenario language is supported by a set of model-based analysis software systems for translating the abstract event conditions to a specific event time range. The event conditions include the desired target state, subsystem interdependency, and subsystem resource requirement. Visualization of the operation scenario execution provides a precise monitoring of the operation behavior of the individual subsystems as well as an integrated system enabling multidisciplinary design validation. Section 6 discusses the observation language syntax, model-based analysis software, and the mission operation visualization.

## 3. Mission Model Framework

A space mission is an extremely complex entity that requires multiple levels of system engineering processes including hardware, software, and operation. Each system engineering process level employs a discipline-oriented framework to organize the development activities. In VM, a mission is regarded from the space science exploration perspective involving science targets, science instruments, and a spacecraft system that operates the science instruments to observe the target phenomena. Thus, the mission model framework is categorized into three domains: a spacecraft system, a payload system, and a target system. The spacecraft system functions include navigation, attitude control, data processing, and communication. The Payload system functions include observation activities of the sensor systems. The target system functions include geometric, radiometric, and dynamic properties of the target phenomena.

As shown in Figure 2, the system properties of the above three domains are grouped into three types: physical, logical, and dynamic system properties. Three model types are composed corresponding to the three property groups: a structure model, physical properties; a performance model, logical properties; and an operation model, dynamic system state properties. Each model type utilizes a structured model description method for automated model object class construction. The composition of the three model types and the model description method are described below.

The subsystems whose operation properties involve mechanical articulation and geometric alignment with the physical world employ the structure model type to describe the structure required for the operation simulation. For example, the structure model of the attitude control subsystem may be a simplified spacecraft structure that can be applied for visual verification of the spacecraft system attitude with respect to the instrument's boresight and sunlight constraints. The structure models are composed based on the CAD (Computer-Aided Design) files of the hardware system. Auxiliary structures, such as Sun direction, instrument's field-of-view, or constraint cones, may be added to indicate the geometric relationship more precisely. Open Inventor language format (IV) is used for 3-D visualization and manipulation of the structure.

A performance model is composed for each subsystem to represent the expected capability ranges of the subsystem functions that are relevant to the operation of the subsystem. For example, the performance model of a science camera system is described for the signal sensitivity and distortion properties by listing the functional capabilities of optics, detectors, and electronics for the relevant geometric and radiometric design specifications. The fidelity of the performance models depends on the required simulation fidelity. In VM, the performance models that are closely related to the science data product generation, such as attitude control, navigation, and instrument, are composed with higher fidelity. The expected capability range of a subsystem may be based on the design specification or the analysis results of a specific design. The inter-subsystem dependency is the main challenge involved in composing the performance models. The inter-dependency is expressed via a set of reserved multidisciplinary model parameters so that the performance models can be cross-referenced.

An operation model is composed for each subsystem representing the general limitations and constraints, and allowed control modes and associated resource usage following section with respect to the software engineering process of the three classes.

## 4. Virtual Prototyping

Implementation of a software system that simulates the properties of a subsystem (prescribed with the three types of models discussed in the previous section) is referred to as "virtual prototyping" in this paper. The term "virtual" is used to indicate that the subsystem exists in cyberspace not in physical space. The term "prototyping" is used to emphasize that the virtual subsystem can be treated as if it were a physical prototype subsystem with respect to commanding, mission data product generation, and resource usage behavior. A virtual prototype is composed of three basic classes: a model configuration class, a static property class, and a dynamic property class. Each class is discussed below.

The model configuration class integrates the three types of model descriptors of a subsystem so that they can be utilized for creating the subsystem properties and operation behavior. The model configuration class is composed of a set of member functions: a model script token analyzer, a model script syntax parser, and several supporting functions. As shown in Figure 3, the C++ codes for the parser and the lexical analyzer are automatically generated by the SAX and LUTHOR software systems [3,4] from the grammar and token descriptions of the model syntax. SAX creates the model configuration object class library integrating the automatically generated member functions (See Figure 3.)

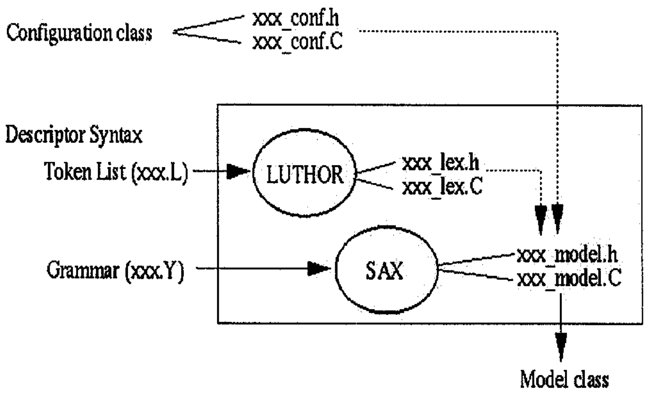


**Figure 3** Automated Model Class Construction

The static property class provides the computational functions necessary to establish the subsystem's identity. The static property is implemented as a derived class of the model configuration class by integrating the software modules that computationally model the subsystem performance. For example, the performance simulator of a camera system is composed of a set of physical property synthesizers representing the camera system component functions (e.g., point spread function, dark current, A/D conversion) and their value specification methods (e.g., uncertainty range, probability distribution).

The dynamic property class is implemented as a friend class of the static property class for simulating time-dependent subsystem states during command execution. The dynamic property class provides two major operation simulation functions, a command handler (subsystem internal state transition manager) and a time-sensitive operation behavior generator (subsystem internal state propagation manager), each of whom is described in the following paragraphs.

The command handler of a subsystem simulates the function of the software module of the subsystem in terms of receiving and verifying the incoming commands, planning the execution of the verified commands, interacting with the hardware devices, and providing the data/information to the external world. For example, the attitude control system has the following five basic commands: Set (deadband, acceleration, velocity), Point (change the attitude to point to a target with a specified body vector), Track (maintain the relative target attitude), Cruise (no turn), and Slew (turn with the specified turn rate). As a specific command is received, the command handler of the attitude control system verifies the command against the current state of the system and plans the execution by consulting with the performance simulator for maximum acceleration, jitter profile, and pointing inaccuracy.

The operation behavior simulator propagates execution of a command after the command handler initiates the command. The propagation of a command execution indicates updating of the spacecraft system state with respect to the external world as well as internal resources as the command execution progresses. During the Point command execution, the attitude of the spacecraft system changes gradually toward the target whose position is estimated by the navigation system. The attitude control system estimates the attitude and generates telemetry stream periodically. Simulation of the operation behavior during a command execution requires simulation of the predicted state, achieved state, and estimated state of the spacecraft system. The three states represent different types of knowledge uncertainty: the predicted state, model uncertainty, the achieved state, performance uncertainty, and the estimated state processing uncertainty.

## 5. Agent-based Integration

The virtual prototype systems described above require multiple layers of coordination in order to perform a desired mission operation as an integrated mission system. The coordination challenges can be grouped into three areas: information exchange among multiple disciplines, command profiles. The general limitations and constraints include subsystem-specific operation peculiarities. For example, a camera system with a specific set of exposure duration settings may list the exposure duration table as a part of the general limitations so that exposure commands can be verified against the available settings in the table. Each control mode is expressed following a predefined command syntax rule that is composed of a command name and a set of parameters. For the automated command syntax verification, each parameter is described with the corresponding data type and range. The resource usage may be specified for the expected execution duration (referenced to a specific computational platform), memory usage, storage system usage, etc.

expressed as a set of low-level parameters when the property has multiple dependencies or involves computational processing of the low-level parameters. The main considerations in defining the model description syntax include subsystem-domain-sensitive model parameter structure for streamlined design product representation, structured expression hierarchy for automated description parsing, and scientifically sensible vocabulary for multidisciplinary communication. These considerations are essential to enable cost effective and timely adaptation of the VML to a specific mission.

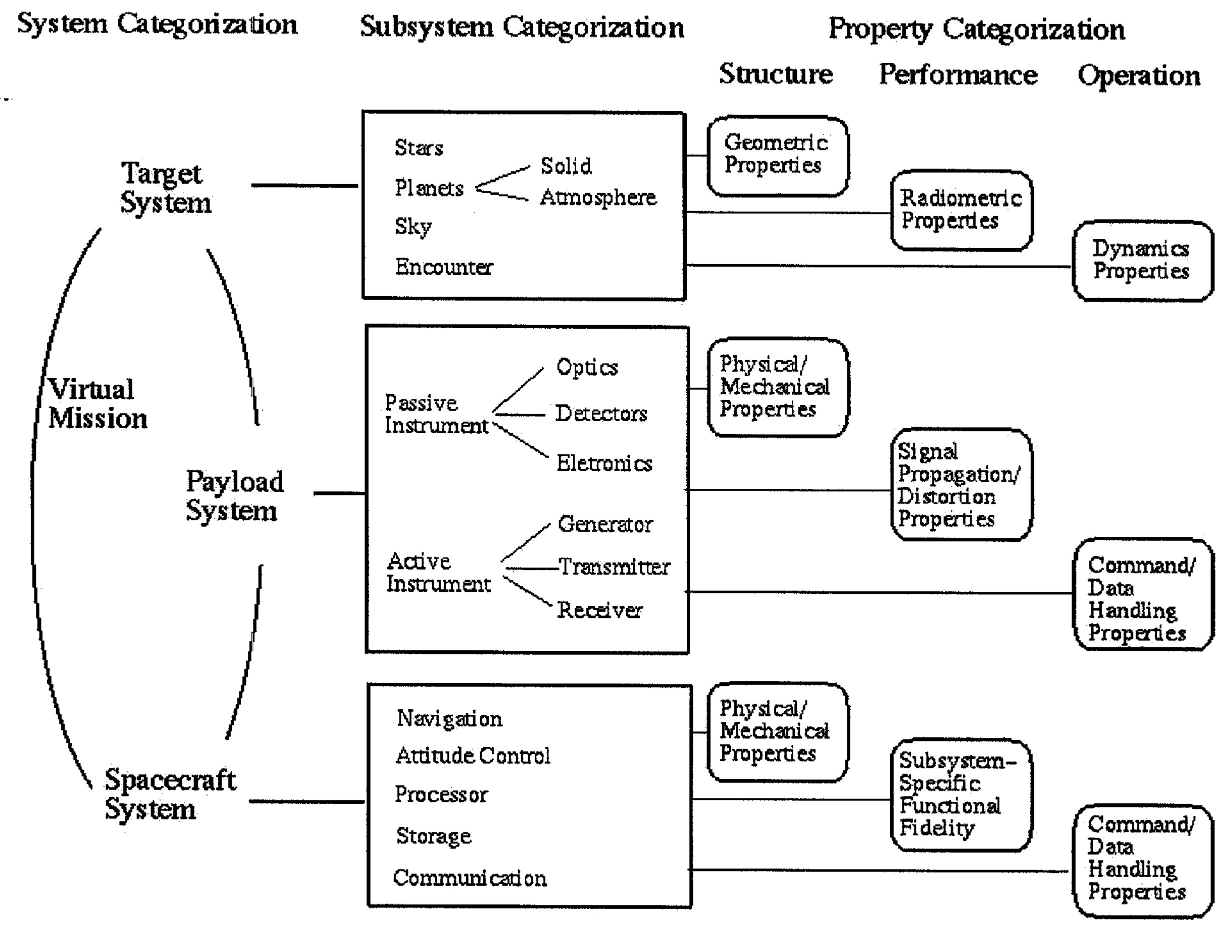


**Figure 2** Mission Model Framework

The three types of models discussed above are composed employing a hierarchical model parameter structure. The basic parameter description syntax is composed of a name and a value where the value can be a scalar, a vector, an array, or a keyword. Each value can be expressed statistically (assuming Gaussian distribution) with an average and a standard deviation. A property may be

The mission model framework enables the simulator of a subsystem to be used as the basis for all cases of the subsystem by loosely coupling the simulator with the model description method. The loose coupling is implemented by creating three object classes: a model configuration class, a static property (operation-mode-independent) class, and a dynamic property (operation-mode-dependent) class. The impact of the mission model framework is discussed in the and data flow among distributed processes, and process distribution among heterogeneous platforms. To overcome the coordination challenges in the above three areas, VM utilizes intelligent mission model agents that are composed of domain-intelligent, platform-independent, and network-friendly information exchange components.

The intelligent agent is capable of understanding and interacting with its environment on the behalf of its user, of moving about the web, and of forming and executing rudimentary decisions. The information agent delivers useful information to the user by actively performing search, access, and retrieval of relevant information. The simulation of the mission design process requires intelligent information agents that search, access, and retrieve design information including spacecraft trajectory, planet ephemerides, planet kinematics, etc.

The combination of datasets from multiple sources is often required to form particular information. Polymorphic functionality is required that can recognize different forms and transform them into a desired product. The technical details of the intelligent mission model agents are presented in the paper "Component-based Implementation of Agents and Brokers for Design Coordination" [5].

As shown in Figure 4, the three systems of the VM integrate the mission system design information through the intelligent mission model agents as well as the locally composed mission models. The locally composed mission models include the models whose content is somewhat static over time (e.g., star catalog) and the models of the systems that are locally designed. The intelligent mission model agents enable integration of the distributed design activities providing a prompt communication of their system-level impact. In this section, the role of three specific mission

The target agent provides information about desired target phenomena. Depending on mission type (fly-by, orbiter, or in-situ), the phenomena of interest and their data resolution vary significantly. For example, during a fly-by mission, the global phenomena of a target (e.g., orbit dynamics, average surface albedo, phase angle) are desired. During the in-situ mission, the local site properties (e.g., rock size distribution, soil mechanics) are desired. Such a wide information range cannot be supported by a single database. Currently, VM employs a PCK (Physics Constant Kernel) agent for the Solar System ephemerides. Planet-specific information agents are being developed for future in-situ missions, including the Mars Program.

The trajectory agent provides the relative state between a target and the spacecraft at a specified time reference system. The state vectors are generated with respect to one reference but are desired with respect to another reference. For example, state vectors are often propagated with respect to the Solar Barycenter during cruise. The position relative to a target then requires translating the vector using the ephemerides of the target. Multiple sources are used for the cruise data and the target ephemerides.

The telecom agent enables dynamic modeling of the communication subsystem during the operation phase. It explores available options by interacting with an on-line telecom performance analysis server, TFP (Telecom Forecast and Predict), and dynamically composes a performance model for the communication subsystem. The automated exploration enables loose coupling between the DSN (Deep Space Network) resource management policy and the mission system design. The complex relationship between antenna performance, DSN service, and spacecraft operation is a major challenge involved in developing the communication subsystem model framework.

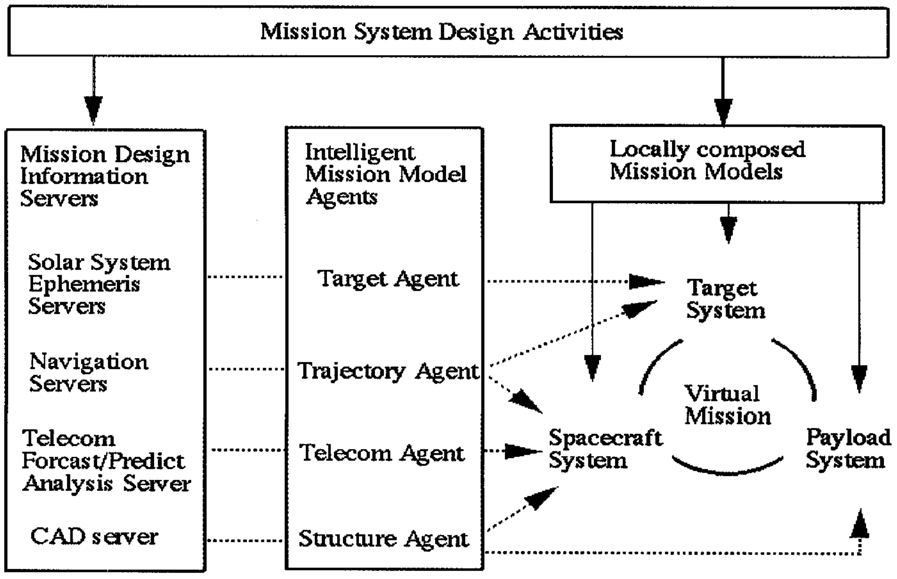

Mission System Design Activities
Mission Design Information Servers
Solar System Ephemeris Servers
Navigation Servers
Telecom Forcast/Predict Analysis Server
CAD server
Intelligent Mission Model Agents
Target Agent
Trajectory Agent
Telecom Agent
Structure Agent
Locally composed Mission Models
Target System
Virtual Mission
Spacecraft System
Payload System

**Figure 4** Intelligent Mission Model Agents

One of the on going mission model agents is the structure agent that will provide multi-resolution structure models by dynamically integrating requested parts for specified polygonal resolutions. The structure agents employ VRML as the middle layer representation format to import various CAD files. These intelligent mission model agents facilitate distributed design information gathering and customization of the information to enable the Virtual Mission Operation phase discussed below.

## 6. Virtual Mission Operation

Virtual Mission Operation refers to execution of science experiment scenarios on the virtual prototypes. The science experiment scenarios represent the target phenomena measurements desired by the mission science team. Traditionally, one science team is organized per science instrument and is responsible for composing the observation sequences in collaboration with the mission operation team during the operation phase of a mission. During the operation phase, the operability of the mission system and the impact of the inter-subsystem dependencies are comprehensively analyzed. If the analysis results are not favorable, it indicates that the science return may be reduced since it will be too late to change the mission system after launch. The ability to perform the operation phase activities during the design phase is critical to achieve a comprehensive design validation and a lifecycle-wide design optimization.

The three major activities of the operation phase activities are command sequence planning, execution (uplink and downlink), and analysis (on board and ground processing). Each activity involves many challenging tasks: the planning activity, multi-discipline optimization, the execution activity, health monitoring; and the analysis activity, science product generation. Performing the above three activities during the design phase requires progressive and automated mechanisms to handle the evolutionary design process and to make up for the lack of the operation team support. Three research tasks in the VM project--Programmable Virtual Mission, Ripples, and Virtual Mission Operation--collectively pursue a progressive and automated operation mechanism. The Programmable Virtual Mission task develops an observation planning language for automated observation sequence planning. The Ripple task develops a Windows-based multi-screen operation console for comprehensive execution monitoring. The Virtual Mission Operation task develops a dynamically configurable operation executive for science product generation by interacting with the virtual prototypes.

*Observation Planning Language*
The observation scenario language, like any language, will have to be evolved over time by the science team, adding more sophisticated expressions. The current syntax is presented in this section not to set the evolutionary path but to initiate the evolutionary process by suggesting an example case. The observation scenario syntax is composed of a list of activities. Each activity consists of an initial condition, a target, and a series of observation events for all subsystems involved in the observation. An observation scenario may be composed of multiple activities. The order of activities is determined based on optimal resource criteria (e.g., total duration, storage usage, down-link time, etc.) of performing all of the activities. The spacing between activities indicates the time involved in turning the spacecraft system from one target position to the next.

The activity target may be described as a specific target name or as a target type with a desired target property. The available target types are Star, Sky, Planet, and Encounter. The target properties are Near/Nearest, Bright/Brightest, Dark/Darkest, etc. The degree of Near, Bright, and Dark can be defined in the scenario. The target types and target properties are provided so that an observation scenario can be composed in an abstract manner. A different target may be chosen for the same scenario depending on the spacecraft system state and the requested observation time range.

The subsystem event is described with an event condition and a subsystem operation command. (See Figure 5.) The event condition is defined as a logical combination of three types of conditions: target condition, time condition, and command condition. The target condition is used to express the necessary target state during the event operation. The supported target states include distance, apparent size, phase angle, etc. The time condition is used to express the required time between events within a subsystem. The command condition is used to express interdependency and concurrency of the events between subsystems.

Two sets of model-based analysis software are developed to support the event-driven scenario analysis: one for the selection of the activity target and target condition of an event, the other for command timing analysis. The first set analyzes each candidate target for the desired condition and verifies the optimality with respect to other mission constraints. For example, the "nearest planet" activity target is selected by computing the turn distance from the initial spacecraft attitude to the nine planets in the solar system. Additional constraint checking is applied for each planet in the order of the turn distance until a planet is found that satisfies the constraints. The command timing analysis is performed for each subsystem based on the operation model of the subsystem. A command may require different execution time depending on the previous commands. For example, the execution time of the "READ ALL" command depends on how many exposures have been commanded prior to the read command. The translated command sequence is submitted to the VM operation executive, mimicking the uplink process. Execution of the command sequence is discussed below. Figure 5 describes the information flow during the model-based observation scenario analysis and command sequence translation process.

articulation viewer updates the spacecraft orientation and instrument articulation, the instrument viewer updates the apparent target in the field of view, and so on.

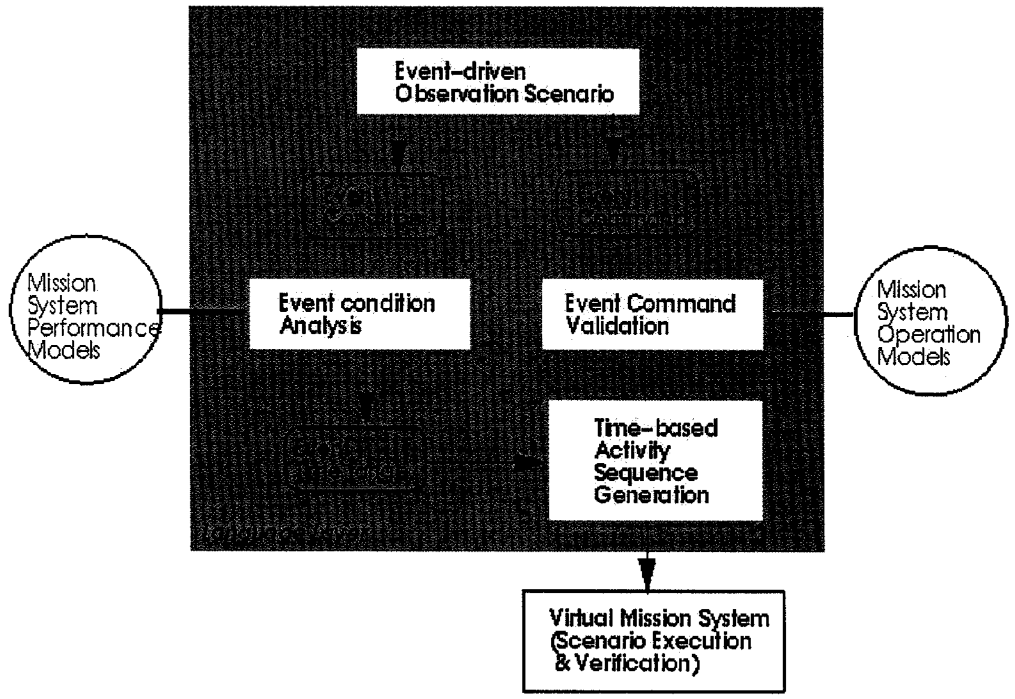

Event-driven Observation Scenario
Mission System Performance Models
Event condition Analysis
Event Command Validation
Mission System Operation Models
Time-based Activity Sequence Generation
Virtual Mission System (Scenario Execution & Verification)

**Figure 5** Observation Scenario Translation

*Scenario Execution & System State Monitoring*

While a spacecraft system is in operation, it generates two types of data products: engineering data and science data. The engineering data captures the state of the spacecraft system, and the science data captures the state of the science targets. The engineering data is also used for processing the science data. The operation team controls the sampling frequency of the engineering data for analysis of desired system behavior. In VM, the engineering data and the science data are directly piped to a multi-screen operation console, Micro-Helm. Micro-Helm is a Windows-based, scalable, distributed visualization server, which is equipped with 9 flat-panel screens forming a 3-by-3-display array. (See Figure 6.)

Micro-Helm has been implemented for monitoring of the simulation process as if it were receiving the telemetry stream from a real spacecraft system. The virtual prototypes generate their operation behavior as time-tagged subsystem states as they execute commands. The time-tagged states are distributed to a set of visualization servers: trajectory, attitude and articulation, instrument, telecom, etc. Each visualization server accesses the structure model of the corresponding subsystem and represents the state in a graphical manner. For example, during the data acquisition operation, the trajectory viewer updates the position of the spacecraft system relative to the target, the attitude and

The time-based operation behavior simulation of the virtual prototype discussed in Section 4 enables this real-time monitoring of the simulation process in collaboration with the VM operation executive [6,7]. The time-based operation behavior simulation is performed with a variable clock where "one second" is defined to be "the time required to simulate one second's worth of operation behavior." One second's worth of operation behavior may take a few milliseconds or several seconds, depending on the computational load of the simulation. To prevent gross simulation time variation, complex computational processes are distributed among multiple processors.

*Science Product Generation*

Science product generation is the ultimate validation of the mission system, and it is one of the most challenging processes in the virtual mission project. Science product generation involves high-fidelity models of a target system, instrument system, and spacecraft system, and requires extensive computation [8]. For example, simulation of an image acquired during the observation of an extended target involves per-pixel ray tracing of the reflected sunlight where the tracing geometry changes as the spacecraft system changes its position and attitude during the exposure duration. The spectral signature of the reflected sunlight also changes depending on the surface material and the spectral sensitivity profile of the instrument. The integration of the

**Figure 6** Virtual Mission and Micro-Helm

Complex geometric, dynamic and radiometric relationships among the Sun, targets, spacecraft system, and instrument provide the operability validation that cannot be acquired with science product generation processes alone.

It is important to note that high-resolution data in the simulation indicates the level of detail in the information, not the accuracy of information. During the design phase, a detailed description may not be made with a high level of certainty. However, if the uncertainty range can be specified, high-resolution measurements can be simulated for the specified uncertainty range, thus providing predicted impacts of the system design on the ultimate science return.

As the mission lifecycle progresses, the uncertainty range may reduce. The provision for high-resolution simulation allows seamless tracking of the reduced uncertainty throughout the lifecycle. In the following section, mission lifecycle alignment is discussed with respect to the extended role of VM at each phase of the lifecycle.

## 7. Mission Lifecycle Alignment

The lifecycle tracking mission system representation is important in three aspects: evaluation of the modeling and simulation technology in the design validation process, inheritance of the validated mission models by the next generation of missions, and extension of the modeling and simulation technology in support of the entire lifecycle. This section addresses the third aspect based on the several years of experience in supporting the JPL Flight System Testbed and science observation planning activity of the Deep Space One mission.

The JPL Flight System Testbed (FST) was established in 1994, as a part of the simulation-based concurrent engineering revolution. The goal of the FST is to create a flight system testbed composed of a set of simulation modules that behave like real subsystems and provide a transparent transition path to real subsystems progressively as they become available. In that way, a subsystem can be tested in an integrated mission system setting. In support of the FST, the VM project develops Instrument Testbed [9] where the instrument-system-related activities--instrument hardware prototyping, calibration software, and flight software-- can be concurrently engineered.

Figure 7 depicts the process of concurrent calibration software development. The development is performed employing a virtual target source and a virtual camera to populate the test database, and a set of the calibration parameters is automatically analyzed from the test database. The accuracy of the analysis software can be easily verified by controlling the quality of the test database through the model parameter setting of the virtual camera. An interesting fact to note is that the performance model of the virtual camera is derived from the calibration parameters. Thus, when the calibration software is applied to the real instrument, the calibration result can be used to update the accuracy of the virtual instrument's model.

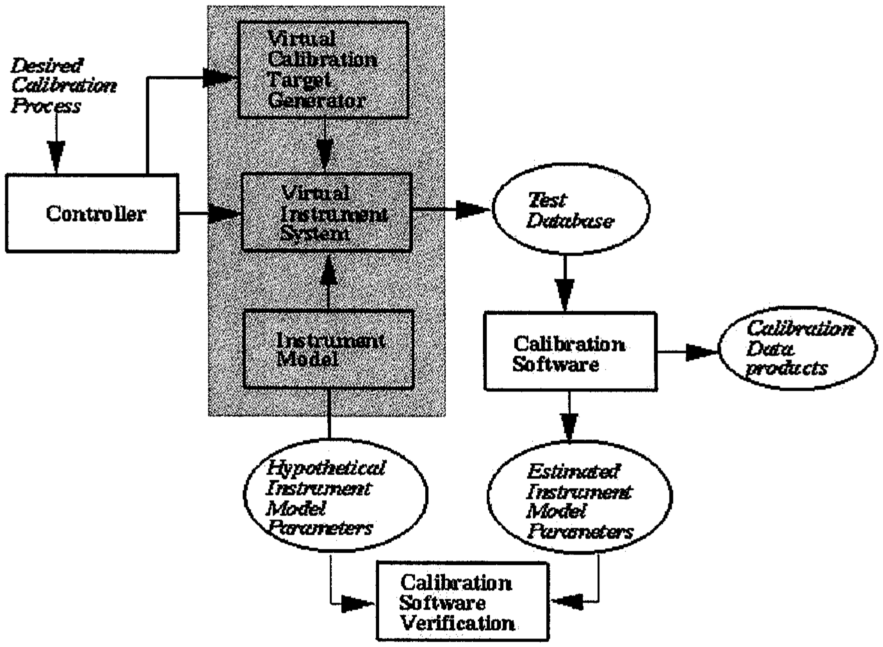

Desired Calibration Process
Virtual Calibration Target Generator
Controller
Virtual Instrument System
Test Database
Instrument Model
Calibration Software
Calibration Data products
Hypothetical Instrument Model Parameters
Estimated Instrument Model Parameters
Calibration Software Verification

**Figure 7** Calibration Software Verification Process

Figure 8 depicts the process of comprehensive flight software validation. In general, flight software development is composed of four parts: instrument command interface, onboard data processing, instrument hardware control, and interaction with the spacecraft system for acquiring the system states necessary for data processing. By integrating the flight software module with virtual instrument hardware and a virtual spacecraft system, the flight software can be comprehensively verified for the ultimate science experiment scenarios.

Figure 9 depicts the process of science observation planning where the VM is viewed as a virtual flight system testbed executing the observation scenario [10]. The experiment design is synthesized by employing the high-level observation scenario design language and the model-based scenario analysis process. The synthesized experiment design is verified on the virtual mission system that simulates the properties of the mission system by executing the VM command sequences. The verified VM command sequence is then translated to a mission-specific operation command sequence following the command dictionary syntax. The science data production capability of VM provides a great advantage over the mission flight system testbed by enabling verification of context-sensitive commands such as data compression, automated target extraction, etc.

## 8. Future Direction

One of the great technical challenges in the mission system design is in creating an intelligent spacecraft system that can explore unknown territories with minimum guidance from ground-based operators. The technical approach toward developing an "intelligent spacecraft system" may be discipline-specific (e.g., automated navigation) or generic (e.g., artificial intelligence). VM approaches it as an evolutionary communication system of multiple discipline-smart subsystems. Therefore, VM seeks to develop a multi-discipline-shared communication framework that enables the evolution path toward the future deep space missions.

The Space Mission Semiotics research activity of VM is developing a shared information categorization structure as the basis of the communication framework [11,12,13,14]. The mission model information categorization structure will be employed to progressively educate the subsystems to

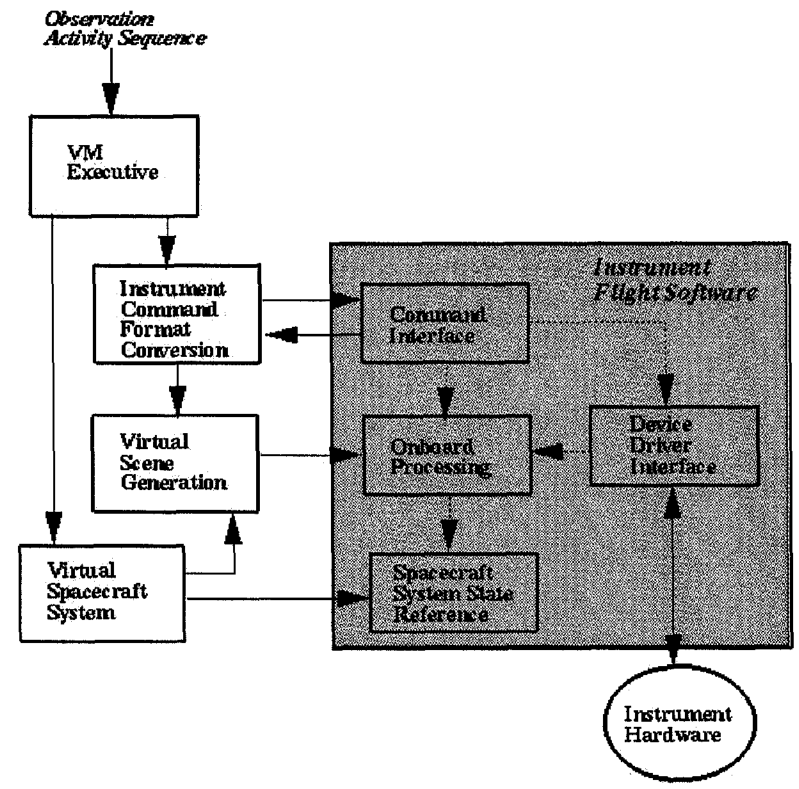

Observation Activity Sequence
VM Executive
Instrument Command Format Conversion
Virtual Scene Generation
Virtual Spacecraft System
Instrument Flight Software
Command Interface
Onboard Processing
Device Driver Interface
Spacecraft System State Reference
Instrument Hardware

**Figure 8** Instrument Flight Software Verification Process

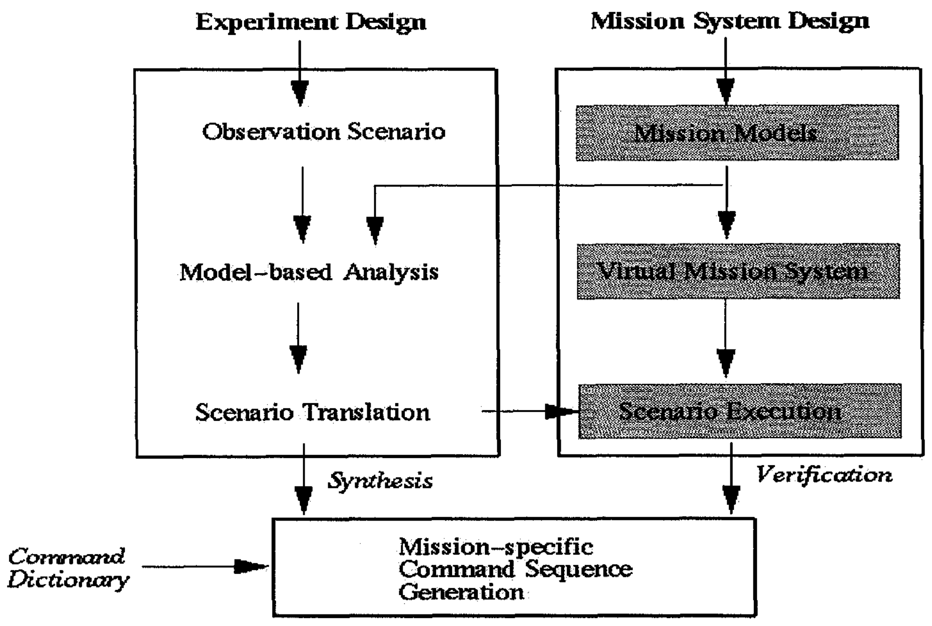


**Figure 9** Virtual Mission Testbed

perform self-diagnosis (self-image formation or self-modeling), to formulate their role with respect to the system (relation modeling), and to plan their operation sequence (autonomy).

The self-image formation is a natural extension of the automated calibration process, which can be achieved by providing a set of calibration target criteria and the target response analysis algorithms. The relation modeling involves each subsystem developing a model of the spacecraft system from the subsystem-centered perspective. This is an extension of the virtual mission system, where multiple projections of the virtual mission system are created, each projection from the point-of-view of a specific subsystem. The autonomous operation planning will be approached as the derivation of the observation scenario language by designing subsystem-centric event conditions and condition analyzers.

## Acknowledgements

The work described in this paper was performed at Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The Information Systems Branch of the Research Program Management Division, Next Generation Infrastructure program, Intelligent Synthesis program, and High Performance Communication and Computation program of NASA supported the research.

## References

[1] M. Lee, R. Weidner, W. Lu, "Mission Lifecycle Modeling and Simulation," 2000 IEEE Aerospace Conference, Big Sky, Montana, March 2000.

[2] K. J. Lieberherr, "Adaptive Object-Oriented Software," An International Thomson Publishing Company, 1996.

[3] R. J. Weidner, *"SAX – Object-oriented parser generator,"* http://cicero.jpl.nasa.gov/~richard/sax.html.

[4] R. J. Weidner, *"LUTHOR – Object-oriented lexical analyzer,"* http://cicero.jpl.nasa.gov/~richard/luthor.html.

CL 98-1095

[5] R. J. Weidner, "A *Component-based Implementation of Agents and Brokers for Design Coordination," submitted to 2001 IEEE Aerospace Conference, Big Sky, Montana, March 2001.*

[6] M. Abajian, *"DEM - Distributed Execution Manager,"* http://msim.jpl.nasa.gov/~abajian/DEM-1.0.html.

bubbles.jpl.nasa.gov CL 00-1829

[7] A. Teng, M. Lee, *"Vmlib: an Object-oriented Library for Visualizing Observation Sequences in Spacecraft Missions,"* Military, Government, and Aerospace Simulation, 1998, Advanced Simulation Technology Conference.

[8] M. Lee, R. Swartz, and R. Weidner, *"SceneGen," NASA* Tech Brief, 1997.

[9] R. J. Weidner, et al., "Instrument Design Laboratory," http://msim.jpl.nasa.gov/IDL.

[10] M. Lee, et al., "MICAS Science Observation," http://msim.jpl.nasa.gov/MICAS.

[11] U. Eco., "A Theory of Semiotics," Indiana University Press, 1976.

[12] M. C. Henson, "Elements of Functional Languages," Blackwell Scientific Publications, 1987.

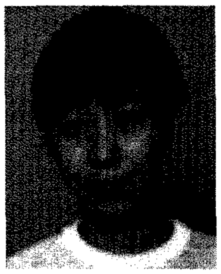
[13] M. Devitt, K. Sterelny, "Language & Reality," A Bradford Book, The MIT Press, Cambridge, Massachusetts, 1995.

[14] G. Fauconnier, "Mappings in Thought and Language," Cambridge University Press, 1997.

*__Meemong Lee__ is a principal technologist at the Jet Propulsion Laboratory. Duirng the last 17 years at JPL, she has developed various Earth and planetary information systems including Spectral Analysis Manager (SPAM), Concurrent Image Processing Executive (CIPE), Planetary Analysis Tool Set (Plato), and Virtual Instrument System (SceneGen). She has been leading the Virtual Mission project since 1996 integrating the modeling and simulation research activities to advance the flight software system and science experiment design and development process. She currently manages three research activities, Programmable Virtual Mission, Virtual Mission System, and In-situ Site Characterization. She also has been supporting the Deep Space 1 mission with the flight software development and science observation sequence design for MICAS instrument system. She has a bachelor's degree in Electronics Engineering from Sogang University in S. Korea, a master's degree in Computer Science and a doctoral degree in Electrical Engineering from Oklahoma State University.*

*__Richard J. Weidner__ is a principal technologist at Jet Propulsion laboratory. During the last 18 years at JPL, he has developed numerous advanced information system technologies and mission operation tools including OOSPICE, QMV, SAX, Luthor, SIMP, SASED, Mars*

*Calendar, Mars Clock, etc. He has been leading the Mission Simulation and Instrument Modeling Group since 1997. He also manages Model-based Design, Intelligent Mission Model Agents, and Ripples activities for supporting the Space Science Information Systems research, Next Generation Infrastructure System, and Intelligent Synthesis Environment Initiative program. He has a bachelor's degree, a master's degree, and a doctoral degree in Electrical Engineering from Oklahoma State University.*



**Wenwen Lu** *is a senior engineer of the Mission Simulation and Instrument Modeling group at Jet Propulsion Laboratory. She has been involved in the Virtual Mission project since she joined the lab in 1997. She leads the Virtual Mission Operation task, developing distributed visualization servers and various analysis software systems for target phenomena modeling and autonomous operation planning. She also supports in-situ Site Characterization activity for the Mars technology program with rock field synthesis and multi-rover-based science instrument operation executive design and development. She has a bachelor's degree from Shanghai University in China, and a doctoral degree in high-energy physics from Caltech.*